Acquisition Overview: The Challenges

Bob Ellison, Software Engineering Institute [vita[1]]
Rita Creel, Software Engineering Institute [vita[2]]

2007-06-04

The challenges of acquiring software-intensive systems continue to grow along with the increasingly critical role software plays in supporting commercial and government enterprise, business, and mission needs. In addition to expanding functionality and complexity, mounting expectations for software systems to be flexible and interoperable add to acquisition challenges, notably in terms of ensuring their security.

Acquisition is, in a sense, outsourcing the development of a system to one or more external providers. This does not relieve the acquirer of responsibility for the outcome. In fact, the activities, products, and behaviors of the acquirer have a significant influence on the success or failure of outsourcing activities. This fact is acknowledged in the appearance of CMMI-based guidelines for outsourcing [Hofmann 06[3]] and the eSourcing Capability Model, a best-practices capability model that has been developed for outsourcing IT-based business functions [Hefley 06[4]]. One purpose of these models is to give client or acquisition organizations guidance on how to improve their own capabilities for participating in outsourcing or acquirer-supplier agreements.

The Acquisition content area of BSI is intended to raise awareness of the acquirer's role in "building security in" for major software-intensive systems. *Assuring Software Systems Security: Life Cycle Considerations for Government Acquisitions*[5] discusses the integration of software security activities into the United States government acquisition life cycle. *Building Security into the Business Acquisition Process*[6] provides an introduction to the standard IEEE 12207, Information Technology – Software life cycle processes, which provides a framework covering the life cycle from conceptualization through retirement [IEEE/EIA 98a[7], 98b[8], 98c[9]].

System complexity and hence acquisition complexity is an aggregate of technology, scale, scope, operational, and organizational issues. For example, consider the initiation phase of the 12207:

1. Prepare a concept or a need to acquire, develop, or enhance a product or service.
2. Prepare a set of requirements including relevant design, testing, and compliance standards.
3. Prepare a risk and cost-benefit analysis for acquisition.
4. Prepare a set of acceptance criteria and criteria for evaluation.
5. Prepare an acquisition plan based on requirements, analyses, and criteria.

1. daisy:208 (Ellison, Robert J.)
2. daisy:888 (Creel, Rita)
3. #hofmann07
4. #hefley06
5. daisy:892 (Assuring Software Systems Security: Life Cycle Considerations for Government Acquisitions)
6. daisy:896 (Building Security into the Business Acquisition Process)
7. #ieee98a
8. #ieee98b
9. #ieee98c

The dynamics of organizational usage increase the complexity of identifying requirements. While elicitation of functional requirements is essential, the primary system architectural drivers are increasingly the non-functional system attributes such as performance, flexibility, and security that enable the desired usage. The acquisition process must explicitly address those non-functional system properties.

In addition, the development of an acquisition plan depends on more than the requirements, the risk and cost-benefit analysis, and the acceptance and evaluation criteria. The plan should be guided by an acquisition strategy that reflects decisions made with respect to a number of strategic issues. Such issues might include acquisition approach (e.g., incremental, evolutionary, agile), external interfaces, use of previously developed or commercial software, competition/solicitation approach, contracting approach, information assurance strategy, training, and support. Strategy drivers impact technical and acquisition requirements, risks, costs, acceptance criteria and approach, and all aspects of operations and sustainment.

Acquisition Challenges

At the forefront of the challenges that strain today’s acquisition practices is the increased scale of deployed systems, where scale is achieved not by building large individual systems but by using network-based integration protocols, such as web services, to integrate a collection of systems. It is software that enables a business work process or a military mission thread to span multiple systems and, increasingly, multiple organizations. The criteria for a successful system deployment includes not only measures for the services associated with that system but also measures for how that system interoperates with other systems to support the work processes or mission threads.

Some of the ways that scale affects acquisition include

- diversity of users
- dynamics of usage
- increased importance of the non-functional requirements

Diversity of Users

Stakeholders who define the business needs for a new application or component can represent an equally complex range of organizational needs; they can be organizationally distributed and diverse, with conflicting, complex, and incomplete requirements. As more organizational functions are linked to share information, the technology that supports these functions is integrated to share data and support cross-functional activities. When the choices made by previously disconnected stakeholders are incompatible, poorly planned integration can leave gaps that provide opportunities for security problems.

Dynamics of Usage

Usage is not static. Today’s systems must support a dynamic operating environment that is driven by evolving business goals and organizational needs. For example, a commercial organization must respond to market changes or regulatory demands. The successful deployment for Department of Defense systems increasingly depends on the ability to securely support very dynamic and networked operational usage that involves multiple and independently developed systems. Both the technologies used and changing operational environment raise security risks that are typically not addressed in current practice.

Systems can also affect usage. New system functionality can change existing work processes, and it is not unusual over time for a system to be used in ways not anticipated in the initial design. Inconsistencies between designed and future usage can be sources for security problems.

As we connect more systems and as that connectivity involves multiple organizations, we have less knowledge of the external dependencies that exist. The multiplicity of factors generates diverse requirements and frequently leads to inherently conflicting ones. There may be requirements that are vague at the start of a development and can only be specified after the dependencies among systems are better understood. Full understanding may come only after deployment and usage. Robust software security practices must be sustained throughout this concurrent evolution of interdependent systems.

Non-Functional Requirements as Architectural Drivers

An increasing number of essential business requirements such as those associated with business continuity or with rapidly changing work processes depend on system quality attributes such as availability, reliability, maintainability, and security. Today's software acquisition efforts focus primarily on functional requirements and capabilities and not on quality attributes.

Most enterprise systems are relatively simple if we just consider the functionality because decades of use have refined the relevant design patterns, but the business drivers have also increased the importance of the non-functional attributes of the systems being acquired. The complexity arises because of the plethora of details regarding rules, policies, and non-functional requirements such as performance and security [Booch 05[10]]. A challenge for role-based access control is dealing with inconsistent use of terms such as manager within a single organization and certainly across multiple organizations. A challenge for software development in general and certainly for acquisition is to capture those details in the acquisition process.

Adaptability/flexibility is a good example of the importance of a non-functional attribute. There will be an increasing need to integrate new capabilities into a system while it is operating. New and different capabilities will be deployed, and unused capabilities will be dropped; the system will be evolving not in phases, but continuously. Software is touted for its flexibility in terms of meeting requirements, but that flexibility is fully available only at the start of development. Design choices to meet specific requirements can constrain other options and limit the ability to make changes after the system is deployed. The integration requirements for a system that must interoperate with existing systems can be dynamic, since the interfaces and usages associated with the existing systems may change before the new system is deployed. This adaptability drives significant requirements for security. Changes in usage or in the underlying technology may change the security threats. In addition, the continuing confrontation between the attacker and defender is dynamic. Improved defenses can lead to the use of new attack patterns to exploit other weaknesses. The analysis of system failures and of the risks of new attack patterns is a continuous activity.

Other trends that raise the importance of the non-functional properties include

- **Systems of systems**: Systems are rarely stand-alone entities. A business work process or a military mission thread depends on integrating multiple systems into systems of systems. Management and operational control of the individual systems are decentralized. Security must now deal with differences in the risks profiles, risk mitigation strategies, and security policies among a collection of systems.
- **Failures:** Attackers often exploit a system's failure to properly manage errors, such as not validating user input. System development has typically concentrated on hardware failures, which in the past were a primary cause of failures. Now, increasingly more catastrophic system failures are the result of software or data input errors. System complexity increases the likelihood of errors that arise from mismatches in the operating assumptions among systems and hence raises the value of good error management for security.
- **System interactions:** Deployment of a new capability that is assembled from existing, operational software components may degrade the performance and security of the overall network of systems.

10. #booch05

Successful deployment depends not only on the behavior of the new capability itself but on its interactions with other components and systems in the environment. The newly deployed capability might generate unexpected contention for shared resources. An attacker may be able to generate a denial of service for a collection of systems by creating behavior on one system that is not tolerated by the other systems. Verification of new capabilities before deployment must consider such security risks.

Summary

Acquirers must clearly identify and describe mission and business needs and both functional and non-functional requirements to prospective suppliers. They must develop an acquisition plan based on a strategy built around key drivers and risks with respect to cost, schedule, performance, and risk. Such drivers include issues of scale and emerging trends in software and system engineering. Acquirers must understand and document the criteria for accepting deliveries and the approach for verifying that the criteria are met.

The acquirer must evaluate candidate suppliers' abilities to deliver the needed capabilities, in the context of both functional and non-functional requirements. Once a supplier is selected, they must execute a well-defined approach to monitor progress toward delivery and to ensure that non-functional requirements, including security, are kept at the forefront throughout the acquisition life cycle.

While the majority of the BSI content targets developers, some material is useful to the acquirer as well. Many of the issues raised in this introduction involve how systems are integrated to provide required capabilities. The Assembly, Integration, and Evolution[11] content area and the more general material in the System Strategies [12]content area provide a more detailed discussion of integration issues and security. The Architectural Risk Analysis[13] content area is applicable to acquisition and the Security Testing [14]content area provides fodder for acceptance and evaluation criteria.

References

[Boehm 06]	Boehm, Barry. "Some Future Trends and Implications for Systems and Software Engineering Processes." *Systems Engineering* 9, 1 (Spring 2006): 1-19.
[Booch 05]	Booch, Grady. Architecture Web Log[15] (2005).
[Hefley 06]	Hefley, William E. & Loesche, Ethel A. *The eSCM-CL v1.1: Model Overview and The eSourcing Capability Model for Client Organizations (eSCM-CL) v1.1*[16]. Information Technology Services Qualification Center, Carnegie Mellon University, 2006.
[Hofmann 07]	Hofmann, Hubert F.; Yedlin, Deborah K.; Mishler,

11. daisy:60 (Assembly, Integration, & Evolution)
12. daisy:878 (System Strategies)
13. daisy:194 (Architectural Risk Analysis)
14. daisy:69 (Security Testing)
15. http://www.booch.com/architecture/blog.jsp
16. http://w3srv.itsqc.cs.cmu.edu/default.aspx

	John W.; & Kushner, Susan. *CMMI(R) for Outsourcing: Guidelines for Software, Systems, and IT Acquisition*. Boston, MA: Addison-Wesley, 2007.
[IEEE/EIA 98a]	IEEE/EIA. *IEEE/EIA 12207.0-1996, Industry Implementation of International Standard ISO/IEC 12207: 1995, ISO/IEC 12207, Standard for Information Technology – Software life cycle processes*. New York: IEEE, March 1998.
[IEEE/EIA 98b]	IEEE/EIA. *IEEE/EIA 12207.0-1996, Industry Implementation of International Standard ISO/IEC 12207: 1995, ISO/IEC 12207, Standard for Information Technology – Software life cycle processes – Life cycle data*. New York: IEEE, April 1998.
[IEEE/EIA 98c]	IEEE/EIA. *IEEE/EIA 12207.2-1997, Industry Implementation of International Standard ISO/IEC 12207: 1995, ISO/IEC 12207, Standard for Information Technology – Software life cycle processes – Implementation considerations*. New York: IEEE, April 1998.
[Maier 06]	Maier, W. Mark. “System and Software Architecture Reconciliation.” *Systems Engineering 9*, 2 (2006): 146-158.
[D&B 00]	Dun & Bradstreet. Press Release about *Barometer of Global Outsourcing* Report[17], February 29, 2000.

Carnegie Mellon Copyright

17. https://www.dnb.com/newsview/0200news7.htm

1. http://www.sei.cmu.edu/about/legal-permissions.html

Fields

Name	Value
Copyright Holder	SEI

Fields

Name	Value
is-content-area-overview	true
Content Areas	Best Practices/Acquisition
SDLC Relevance	Cross-Cutting
Workflow State	Publishable